OR 2018 OPERATIONS RESEARCH BRUSSELS

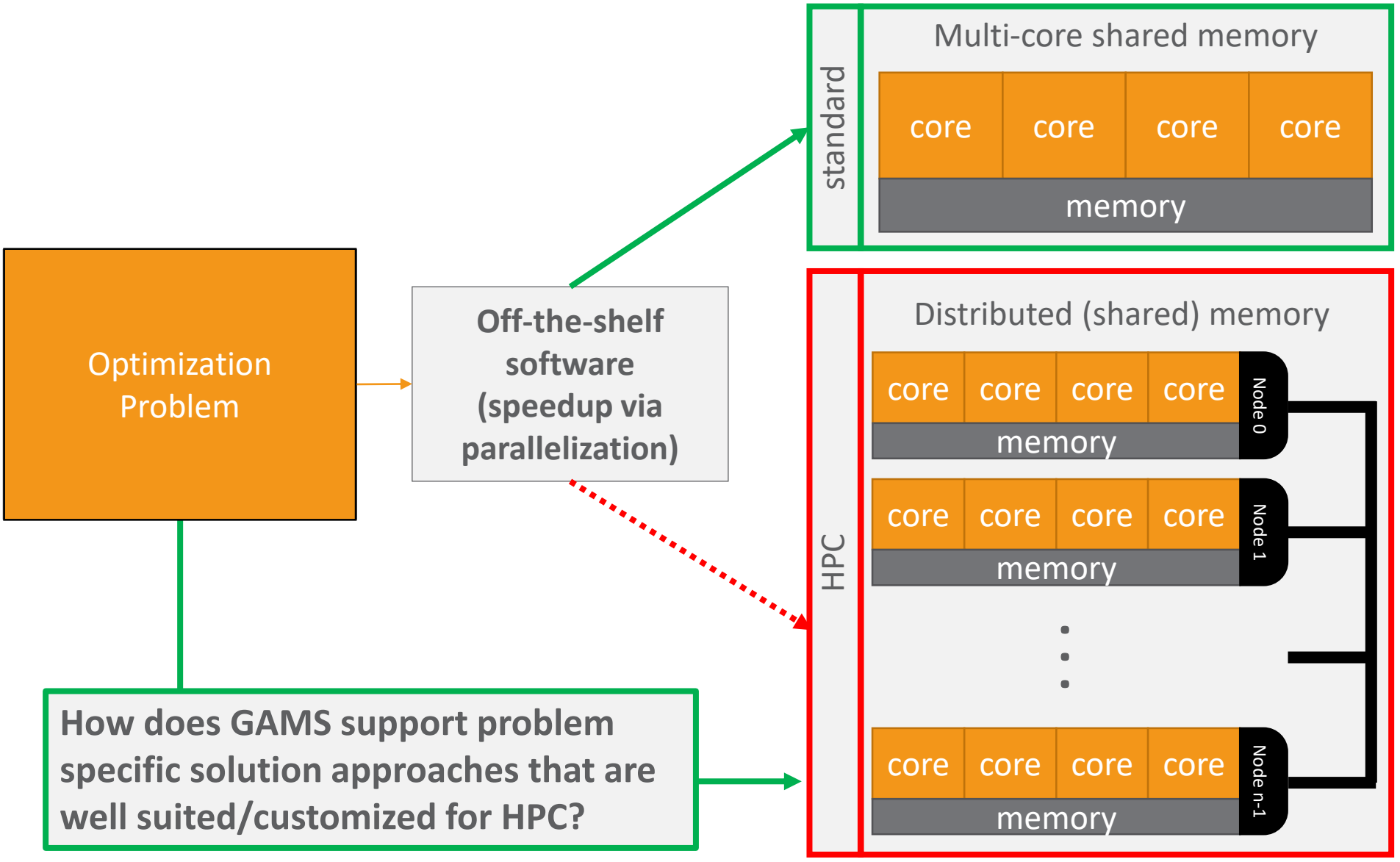
GAMS

GAMS and High-Performance Computing

Frederik Fiand
Operations Research Analyst,
GAMS Software

Motivation

Available Computing Resources

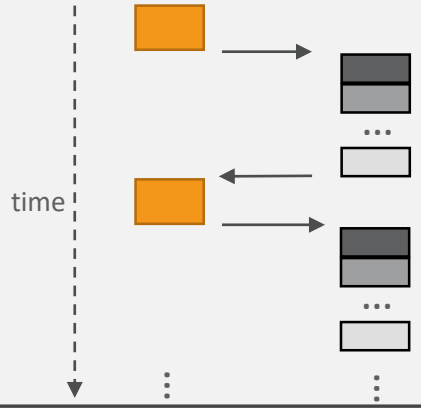


1 Decompose

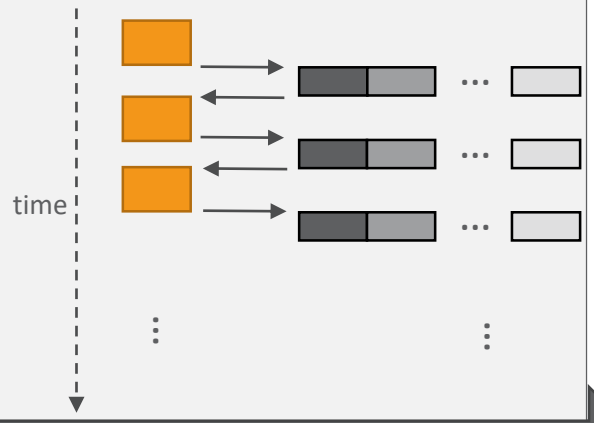
Optimization Problem

2 Use structure exploiting solver

Solve sequence of problems (e.g. in Benders Decomp.)



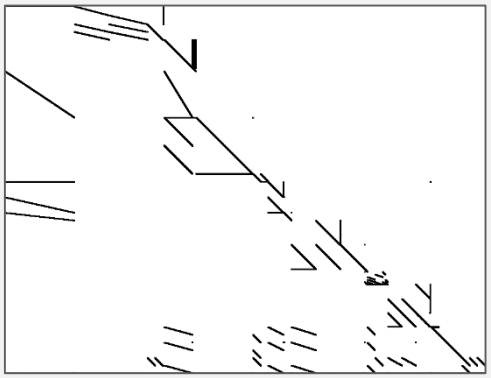
Parallelize



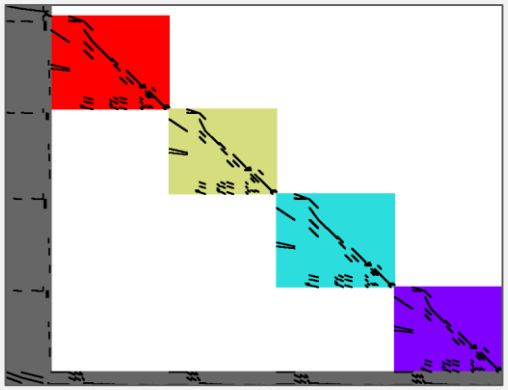
GAMS supports different levels of parallelization

GAMS provides annotation facilities and HPC solver links

Non-zero Plot of matrix



User Annotation to define block structure



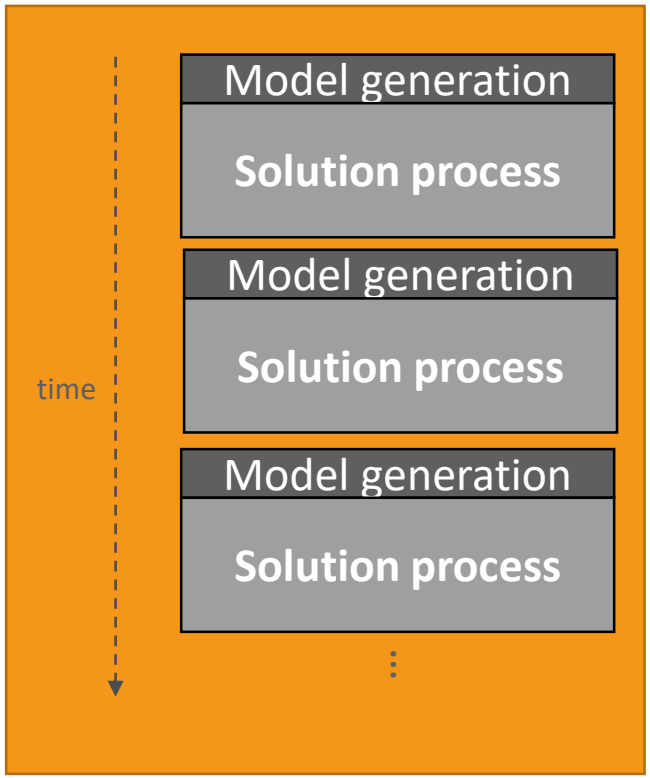
Parallelization with GAMS

From simple sequential to highly parallel solve statements

Sequential Solve Statements in Loops

- loop body code in sequence, often with an expensive solve statement:

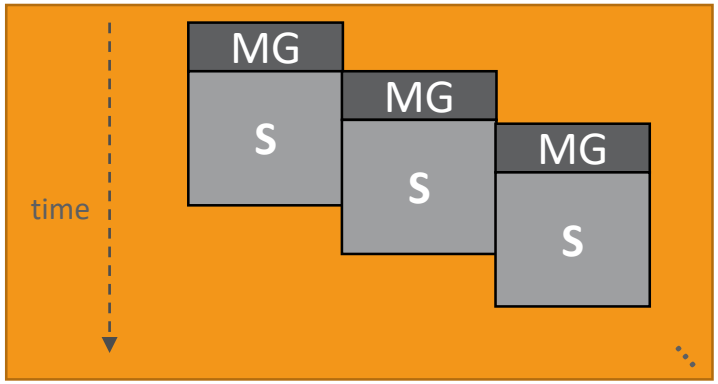
```
... // preparatory work
loop(scen,
    ... // setup model
    option clear=s; s(scen) = yes;
    solve mymodel min obj using minlp;
    ... // store results
);
... // reporting
```



Parallel Solves – GAMS Grid Facility

- SolveLink option specifies the solver linking conventions
- Split loop in submission & collection loop:

```
... // preparatory work
parameter h(scen); mymodel.solvelink=%solvelink.async...%;
loop(scen,
    ... // setup model
    option clear=s; s(scen) = yes;
    solve mymodel min obj using minlp;
    h(scen) = mymodel.handle;
);
repeat
    loop(scen$handlecollect(h(scen)),
        ... // store results
        h(scen) = 0;
    );
until card(h)=0;
... // reporting
```



- Model generation and loop body code in sequence
- Either file based IO or limited to shared memory

Excursus 1: Message Passing Interface (MPI)

```
mpiexec -n 5 gams myfile
```

```
gams myfile
gams myfile
gams myfile
gams myfile
gams myfile
```

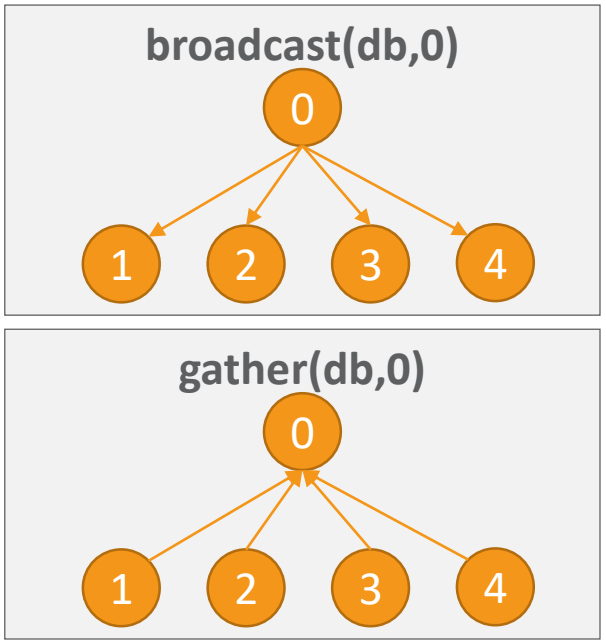
Excursus 1: Message Passing Interface (MPI)

```
mpiexec -n 5 gams myfile
```

```
gams myfile <-- %sysenv.PMI_RANK%=0
gams myfile <-- %sysenv.PMI_RANK%=1
gams myfile <-- %sysenv.PMI_RANK%=2
gams myfile <-- %sysenv.PMI_RANK%=3
gams myfile <-- %sysenv.PMI_RANK%=4
```

myfile.gms – pseudo code

```
$ifthen %sysEnv.PMIRANK%==0
... // preparatory work
$endif
abort$broadcast(db,0) 'problem with broadcast';
... // setup model
s(scen) = ord(scen)-1=%sysEnv.PMIRANK%;
solve mymodel min obj using minlp;
... // store results
abort$gather(db,0) 'problem with gather';
);
$ifthen %sysEnv.PMIRANK%==0
... // reporting
$endif
```



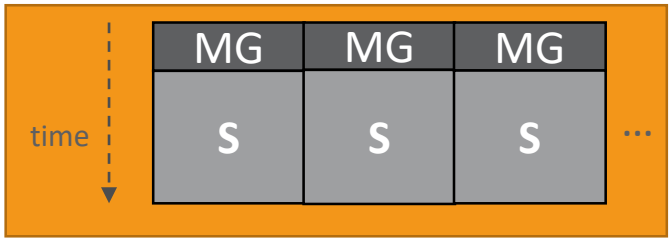
Excursus 1: Message Passing Interface (MPI)

```
mpiexec -n 5 gams myfile
```

```
gams myfile <-- %sysenv.PMI_RANK%=0
gams myfile <-- %sysenv.PMI_RANK%=1
gams myfile <-- %sysenv.PMI_RANK%=2
gams myfile <-- %sysenv.PMI_RANK%=3
gams myfile <-- %sysenv.PMI_RANK%=4
```

myfile.gms – pseudo code

```
$ifthen %sysEnv.PMIRANK%==0
... // preparatory work
$endif
abort$broadcast(db,0) 'problem with broadcast';
... // setup model
s(scen) = ord(scen)-1=%sysEnv.PMIRANK%;
solve mymodel min obj using minlp;
... // store results
abort$gather(db,0) 'problem with gather';
);
$ifthen %sysEnv.PMIRANK%==0
... // reporting
$endif
```



Excursus 1: Message Passing Interface (MPI)

```
mpiexec –n 5 gams myfile
```

```
gams myfile <-- %sysenv.PMI_RANK%=0
gams myfile <-- %sysenv.PMI_RANK%=1
gams myfile <-- %sysenv.PMI_RANK%=2
gams myfile <-- %sysenv.PMI_RANK%=3
gams myfile <-- %sysenv.PMI_RANK%=4
```

myfile.gms – pseudo code

```
$ifthen %sysEnv.PMIRANK%==0
... // preparatory work
$endif
abort$broadcast(db,0) 'problem with broadcast';
... // setup model
s(scen) = ord(scen)-1=%sysEnv.PMIRANK%;
solve mymodel min obj using minlp;
... // store results
abort$gather(db,0) 'problem with gather';
);
$ifthen %sysEnv.PMIRANK%==0
... // reporting
$endif
```

- Requires reorganization of the code but allows parallel solve
- Distribute/merging data easy (part of MPI)
- Network based communication
- Need to make GAMS aware of MPI → **Embedded Code**

Excursus 2: Embedded Code Facility

24.9.1 Major release (August 30, 2017)

Acknowledgments

We would like to thank all of our users who have reported problems and made suggestions for improving this release. In particular, we thank Etienne Ayotte-Sauvé, Wolfgang Britz, Florian Habermacher, Florian Häberlein, Maximilian Held, Ignacio Herrero, Hanspeter Höschle, Erwin Kalvelagen, Toni Lastusilta, John Ross, and Tom

GAMS System

GAMS

- New feature, the **Embedded Code Facility**: This extends the connectivity of GAMS to other programming languages. It allows the use of Python code during compile and execution time. GAMS symbols are shared with the external code, so no communication via disk is necessary.
  The embedded code feature is available on Linux, MacOS X, and Windows. For these platforms, a Python 3.6 installation is included with the GAMS distribution. If the user wants to work with a different Python 3.6, installed separately, for models with embedded code the new command line option **pySetup** needs to be set to 0.

  Note

  This feature is currently in beta status. Any feedback to support@gams.com is appreciated.
- New command line option **procDirPath**: Specifies the directory where the process directory should be created.

Example: Sequential Benders Decomposition

```
set scen     'scenario set' / scen1*scen100 /
    s(scen)  'dynamic secenario subset'
    k        'benders iterations' / k1*k1000 /;

... // preparatory work
loop(k$( NOT done ),
  ... // setup model for master-problem
  solve master min obj_master use lp;
  ... // fix first stage variables
  loop(scen,
    ... // setup model for sub-problem
    option clear=s; s(scen) = yes;
    solve sub min obj_sub use lp;
    ... // process results
  );
  ... // compute cuts for next master
  ... // free fixed first stage variables
  ... // set done=1 if convergence criterion is met
);
... // reporting
```

Example: Parallel Benders with mpi4py

PMI_RANK=0

```
set scen     'scenario set' / scen1*scen100 /
    s(scen)  'dynamic secenario subset'
    k        'benders iterations' / k1*k1000 /;

...
embeddedCode Python:
  from mpi4py import *
  comm = MPI.COMM_WORLD
  ...
pauseEmbeddedCode
... // preparatory work
$ifthen.MPI 0==%sysenv.PMI_RANK%
loop(k$( NOT done ),
  ... // setup model for master-problem
  solve master min obj_master use lp;
  ... // fix first stage variables
  continueEmbeddedCode:
    comm.bcast([[done]] + <data for sub>, root=0)
    cut = comm.gather(None, root=0)[1:]
    ... // gathered data → GAMS data struct.
  pauseEmbeddedCode <load GAMS data struct.>
  ... // compute cuts
  ... // free fixed first stage variables
  ... // set done=1 if convergence criterion is met
);
continueEmbeddedCode:
  comm.bcast([[done],<empty>], root=0)
endEmbeddedCode
... // reporting
$else.MPI
```

PMI_RANK>=1

```
set scen     'scenario set' / scen1*scen100 /
    s(scen)  'dynamic secenario subset'
    k        'benders iterations' / k1*k1000 /;

...
embeddedCode Python:
  from mpi4py import *
  comm = MPI.COMM_WORLD
  ...
pauseEmbeddedCode
... // preparatory work
$else.MPI
  s(scen) = ord(scen)=%sysenv.PMI_RANK%;
  while(1,
    continueEmbeddedCode:
      primal_solution = comm.bcast(None, root=0)
      // broadcasted data → GAMS data struct.
    pauseEmbeddedCode <GAMS data struct.>
    abort.noerror$done 'terminating subprocess';
    solve sub min obj_sub use lp;
    ... // process results
    continueEmbeddedCode:
      comm.gather(<subproblem results>), root=0 )
    pauseEmbeddedCode
  );
$endif.MPI
```

Computational Result(s)

- Two-stage stochastic problem emerged from energy system model
- 100 scenarios
- Deterministic Equivalent:
  21,029,101 rows, 23,217,077 columns, 85,721,477 non-zeroes
- Benders:
  - Master: up to 553 rows, 177 columns, 24,911 non-zeroes
  - Sub: 210,282 rows 232,161 columns 696,461 non-zeroes
  - 19 lines of Python Code + some refactorization of GAMS code for MPI version

	TIME [sec]		
Method	**sub-problems**	**master-problem**	**total**
Deterministic Equivalent[1]			4059.00
Seq. Benders[2]	2394.92	0.18	2395.10
MPI Benders[3]	28.35	0.16	28.51

All runs were made with GAMS 25.1.2 on JURECA@JSC with 24 cores per node, 2.5 GHz, (Intel Xeon E5-2680 v3 Haswell), 128 GB RAM

1: single node, 16 cores, CPLEX barrier, no crossover

2: single node, 4 cores per solve statement, CPLEX barrier, advind 0

3: 17 nodes, 404 cores in total, 4 cores per solve statement, CPLEX barrier, advind 0

Structure exploiting HPC Solvers

Annotation Facilities and the GAMS/PIPS-IPM-link

Project BEAM-ME

Supported by:

Federal Ministry
for Economic Affairs
and Energy

on the basis of a decision
by the German Bundestag

Implementation of acceleration strategies from mathematics and computational sciences for optimizing energy system models.

Realisierung von Beschleunigungsstrategien der anwendungsorientierten Mathematik und Informatik für optimierende Energiesystemmodelle.

A PROJECT BY

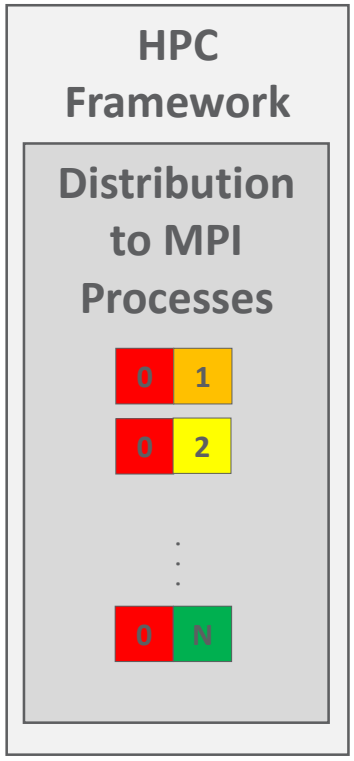
FORSCHUNGSZENTRUM JÜLICH

DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

PIPS-IPM[1]

Consider LP with block-diagonal structure, linking constraints, and linking variables (the kind of problem we want to solve):

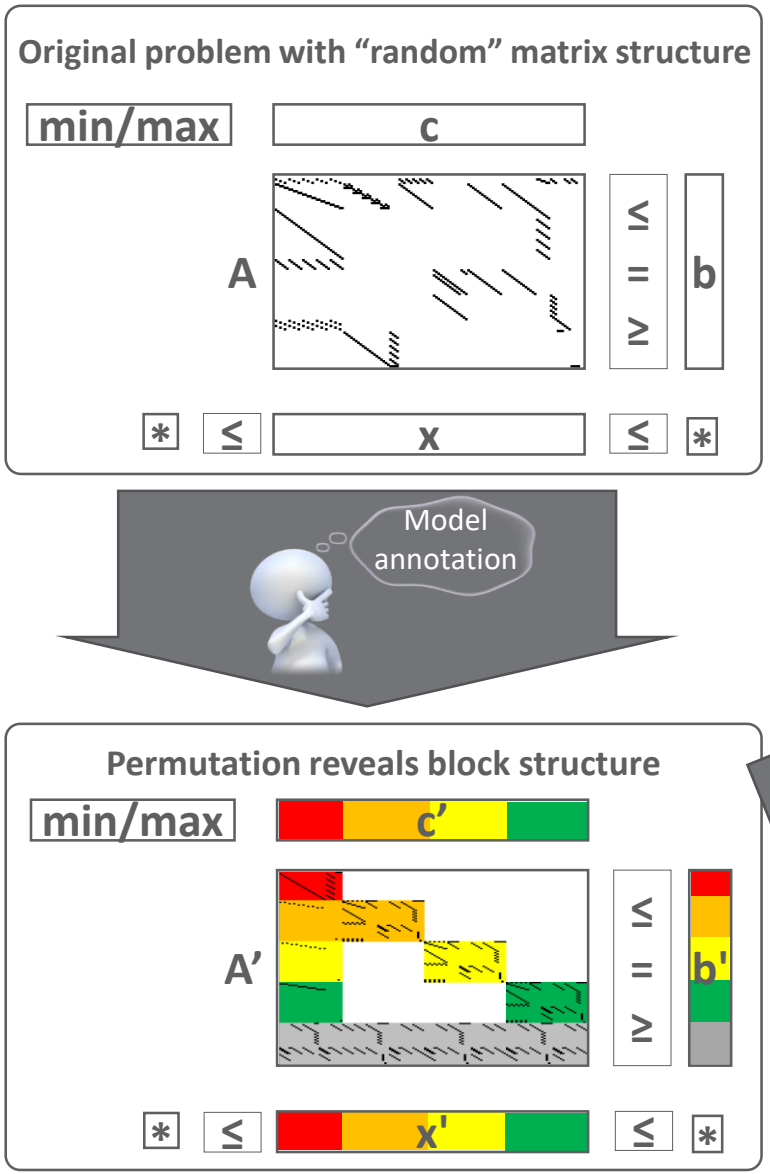
$$
\begin{aligned}
\min \quad & \sum_{i=0}^{N} c_i^T x_i \\
\text{s.t.} \quad & T_0 x_0 \\
& T_1 x_0 + W_1 x_1 \\
& T_2 x_0 + \qquad W_2 x_2 \\
& \vdots \qquad\qquad\qquad \ddots \\
& T_N x_0 + \qquad\qquad\qquad W_N x_N \\
& F_0 x_0 + F_1 x_1 + F_2 x_2 \cdots F_N x_N
\end{aligned}
\qquad
\begin{aligned}
& \\
& = b \\
& = h_1 \\
& = h_2 \\
& \vdots \\
& = h_N \\
& = g
\end{aligned}
$$



- Block diagonal structure allows parallelization of linear algebra within PIPS-IPM
- Solve *N* systems of linear equations in parallel instead of one huge system

[1] Petra et al. 2014: *"Real-Time Stochastic Optimization of Complex Energy Systems on High-Performance Computers"*

GAMS/PIPS-IPM Solver Link - Overview



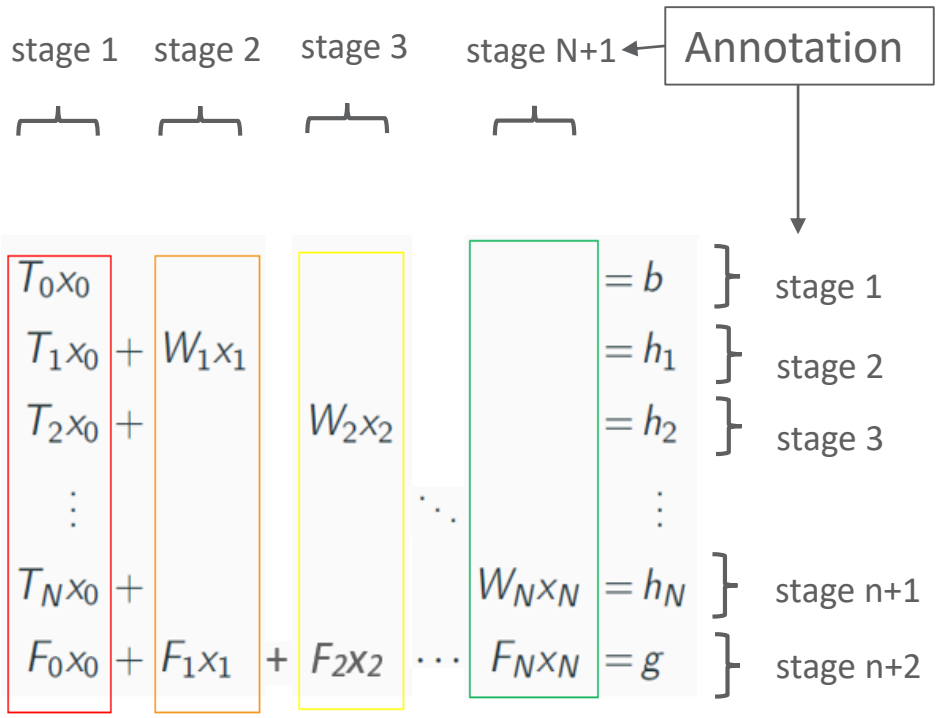
$$\min \sum_{i=0}^{N} c_i^T x_i$$

$$\text{s.t.} \begin{bmatrix} T_0 x_0 \\ T_1 x_0 \\ T_2 x_0 \\ \vdots \\ T_N x_0 \\ F_0 x_0 \end{bmatrix} + \begin{bmatrix} W_1 x_1 \\ \\ \\ \\ \\ F_1 x_1 \end{bmatrix} + \begin{bmatrix} W_2 x_2 \\ \\ \\ \\ \\ F_2 x_2 \end{bmatrix} + \begin{bmatrix} \\ \\ \\ \ddots \\ \\ \cdots \end{bmatrix} \begin{bmatrix} \\ \\ \\ \\ W_N x_N \\ F_N x_N \end{bmatrix} = \begin{bmatrix} b \\ h_1 \\ h_2 \\ \vdots \\ h_N \\ g \end{bmatrix}$$

Model Annotation

Model Annotation by .stage attribute

Matrix structure required by PIPS API

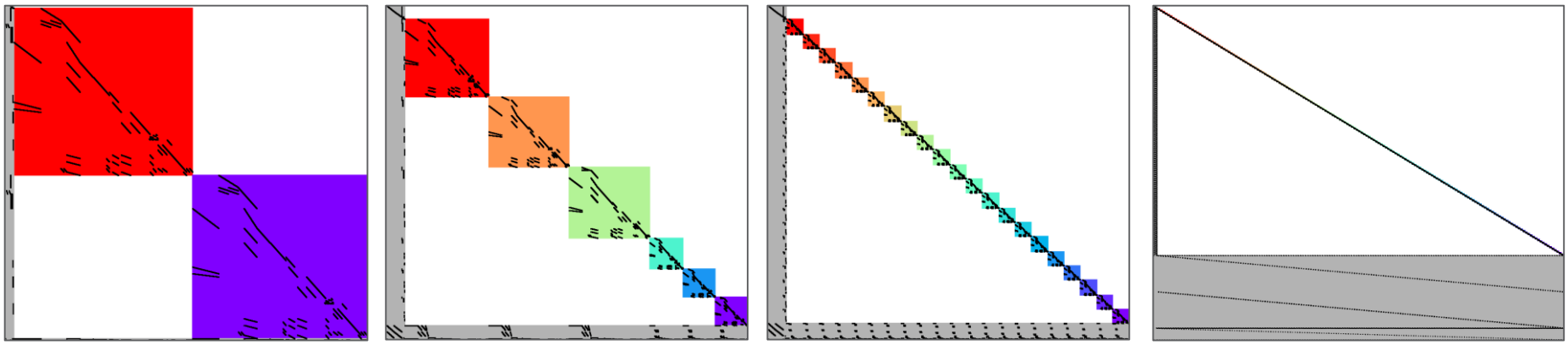


Exemplary Annotation for simple energy system model (regional decomposition)

```
[...]
* Master variables and equation
FLOW.stage(t,net(rr1,rr2))        = 1;
LINK_ADD_CAP.stage(net(rr1,rr2)) = 1;
[...]
* Block variables and equations
POWER.stage(t,rp(rr,p))     = ord(rr)+1;
EMISSION_SPLIT.stage(rr,e) = ord(rr)+1;
[...]
eq_power_balance.stage(t,rr)    = ord(rr)+1;
eq_emission_region.stage(rr,e) = ord(rr)+1;
eq_emission_cost.stage(rr,e)    = ord(rr)+1;
[...]
* Linking Equation
eq_emission_cap.stage(e) = card(rr)+2;
```

Model Annotation cont.

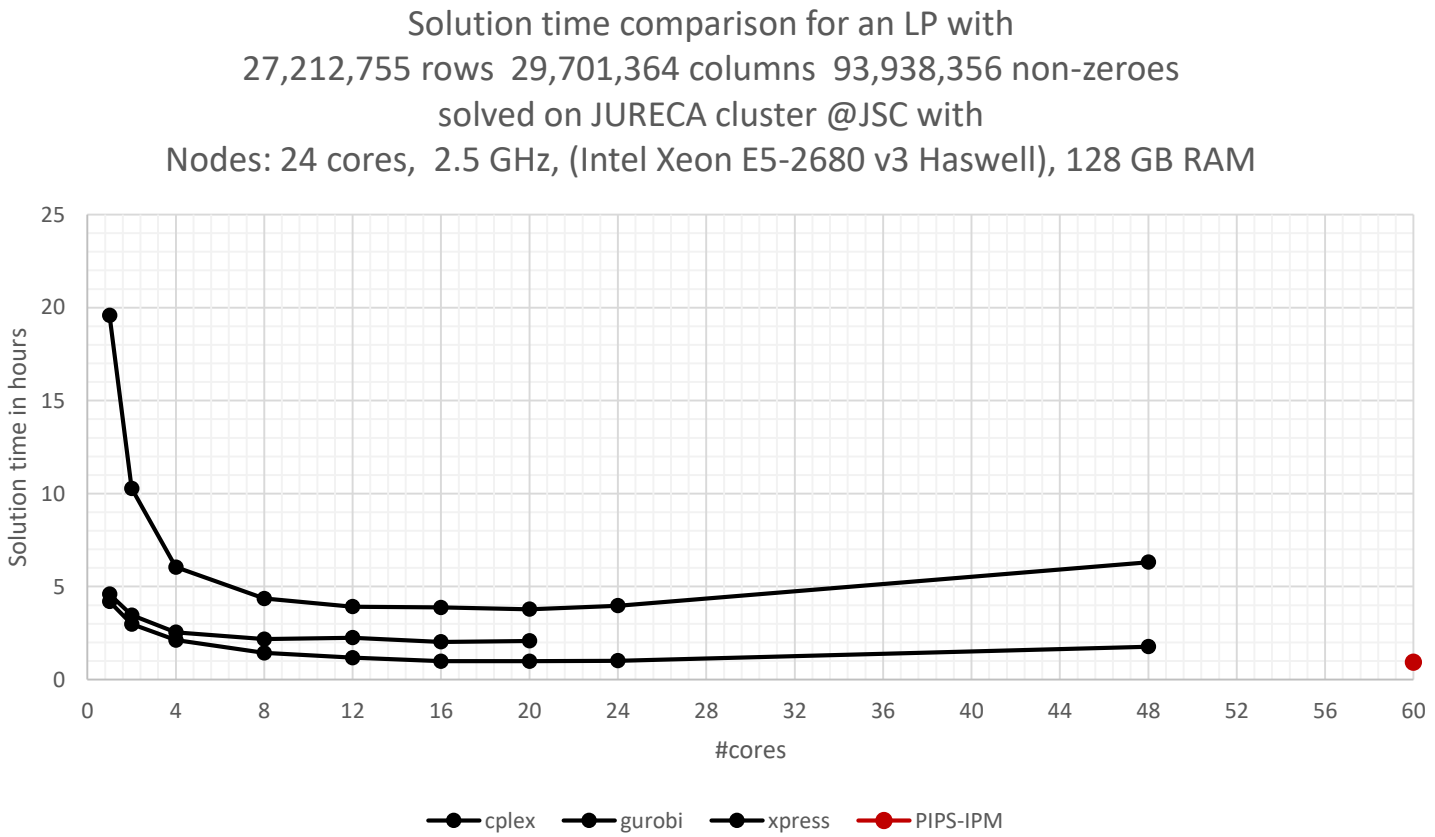
- How to annotate Model depends on how the model should be “decomposed” (by region, time,…)



Plots show four different annotations of identical model

- Blocks of equal size are beneficial

Computational Result(s)



Summary / Outlook

Summary

- GAMS provides broad set of parallelization facilities
- HPC Capabilities of GAMS can be easily extended via embedded Python Code (Parallel Benders with mpi4py)
- Annotation Facilities to allow users the definition of block structures are available
- Link to HPC solver PIPS-IPM available

Outlook

- Embedded Python Code in combination with GAMS Python OO API allows to further increase efficiency (e.g. via GAMS ModelInstances, Warmstarts, …)
- Additional HPC Solver Link to OOPS[1,2] is currently under development
- Parallelization can be extended to Model Generation
  - Usual Model": model generation time << solver time
  - For LARGE-scale models the model generation may become significant:
    - due to time consumption
    - due to memory consumption
    - due to hard coded limitations of model size (# non-zeroes < ~2.1e9)
  - Generation of separate model blocks as required by solver
    - Fully implemented by user: possible (significant refactorization of code)
    - Annotation provided by user → block sharp generation by GAMS: work in progress

[1]: **J. Gondzio and R. Sarkissian**, Parallel Interior Point Solver for Structured Linear Programs, *Mathematical Programming* **96** (2003) No 3, 561-584.
[2]: **J. Gondzio and A. Grothey**, Reoptimization with the Primal-Dual Interior Point Method, *SIAM Journal on Optimization* **13** (2003) No 3, pp. 842-864.

OPERATIONS RESEARCH 2018 BRUSSELS

GAMS

Frederik Fiand

Operations Research Analyst, GAMS Software

ffiand@gams.com